تعلم لغة:

p h p

المادة مترجمة من موقع:

W3schools

PHP

هي لغة برمجة نصية للخوادم، وأداة قوية لإنشاء صفحات ويب حيوية وتفاعلية، وهي بديل واسع الاستخدام ومجاني وفعال للمنافسين مثل ASP من Microsoft.

PHP 7 هو أحدث إصدار مستقر.

مــثـــال:

```
<!DOCTYPE html>
<html>
<body>

<?php
echo "My first PHP script!";
 ?>

</body>
</html>
```

-----(((مقدمة PHP)))-----

تستخدم أكواد PHP التنفيدية للخوادم.

ما يجب أن تعرفه:

قبل المتابعة، يجب أن يكون لديك فهم أساسي لما يلي:

- HTML
- CSS
- JavaScript

ما هي PHP؟

- **PHP** هي اختصار لـ "PHP: Hypertext Preprocessor".
- **PHP** هي لغة برمجة مفتوحة المصدر تستخدم على نطاق واسع.
- يتم تنفيذ البرامج النصية لـ PHP على الخوادم.
- **PHP** مجانية للتنزيل والاستخدام.

PHP هي لغة قوية ومشهورة!

أنها قوية بما يكفي لتكون في صميم أكبر نظام تدوين على شبكات الإنترنت (WordPress)!

إنها عميقة بما يكفي لتشغيل أكبر شبكة اجتماعية (Facebook)!

من السهل أيضًا أن تكون لغة الخادم للمبتدئين!

ما هو ملف PHP؟

يمكن أن تحتوي ملفات PHP على نصوص وأكواد: HTML، CSS، JavaScript، PHP.

يتم تنفيذ أكواد PHP على الخوادم، ويتم إرجاع النتيجة إلى المتصفح كـ HTML عادي.

ملفات PHP لها امتداد ".php".

ماذا يمكن أن تفعل PHP؟

- ❖ إنشاء محتوى حيوي لصفحات الإنترنت
- ❖ إنشاء وفتح وقراءة وكتابة وحذف وإغلاق الملفات على الخوادم
- ❖ جمع بيانات النموذج
- ❖ إرسال واستقبال ملفات تعريف الارتباط
- ❖ إضافة أو حذف أو تعديل البيانات في قاعدة البيانات الخاصة بك
- ❖ التحكم في وصول المستخدم
- ❖ تشفير البيانات
- ❖ لا تقتصر PHP على إخراج صفحات HTML فقط. بل يمكنها إخراج الصور وملفات PDF. يمكنها أيضًا إخراج أي نص، مثل XHTML وXML.

لماذا PHP؟

- ❖ PHP تعمل على منصات متعددة (Windows،Linux،Unix،Mac OS X، إلخ).
- ❖ PHP تتوافق مع جميع الخوادم المستخدمة حاليًا (Apache،IIS... إلخ).
- ❖ PHP تدعم مجموعة واسعة من قواعد البيانات.
- ❖ PHP مجانية. قم بتنزيلها من موقع PHP الرسمي: www.php.net.
- ❖ PHP سهلة التعلم وتعمل بكفاءة من جانب الخادم.

ما الجديد في PHP 7

- ❖ PHP 7 أسرع بكثير من الإصدار الثابت السابق المعروف (PHP 5.6).
- ❖ تدعم PHP 7 تعريفات أكثر صرامة لمدخلات الدوال.
- ❖ تدعم PHP 7 العمليات الجديدة (مثل عملية السفينة الفضاء: <=>).

تثبيت PHP

ماذا أحتاج؟

لبدء استخدام PHP، يمكنك:

- البحث عن مضيف ويب يدعم PHP وMySQL.
- قم بتثبيت خادم ويب على حاسوبك، ثم قم بتثبيت PHP و MySQL.

استخدم مضيف ويب يدعم PHP

إذا قام خادمك بتنشيط دعم PHP، فلن تحتاج إلى القيام بأي شيء.

ما عليك سوى إنشاء بعض ملفات php.، ووضعها في مجلد الويب الخاص بك، وسيقوم الخادم تلقائيًا بتحليلها لك.

لا تحتاج إلى ترجمة أي شيء أو تثبيت أي أدوات إضافية.

نظرًا لأن PHP مجانية، تقدم معظم مضيفات الويب دعم PHP.

قم بإعداد PHP على الحاسوب الخاص بك

ومع ذلك، إذا كان خادمك لا يدعم PHP، فيجب عليك:

- تثبيت خادم الويب.
- تثبيت PHP.
- تثبيت قاعدة بيانات، مثل: MySQL.

يحتوي موقع PHP الرسمي (PHP.net) على تعليمات التثبيت:

http://php.net/manual/ar/install.php

بنيـة جـمـل PHP

يتم تنفيذ البرنامج النصي لـ PHP على الخوادم، ويتم إرسال نتيجة HTML العادية مرة أخرى إلى المتصفح.

بنية الجملة الأساسية لـ PHP

يمكن وضع نص PHP في أي مكان في المستند.

يبدأ نص PHP بـ <?php وينتهي بـ ?>:

```
<?php
// ضع أكواد php هنا
 ?>
```

الامتداد الافتراضي لملفات PHP هو ".php".

يحتوي ملف PHP عادةً على أوسمة HTML، وبعض التعليمات النصية البرمجية لـ PHP.

لدينا مثال بسيط لملف PHP، مع برنامج PHP نصي يستخدم دالة PHP مدمجة "echo" لإخراج نص "Hello World!" على صفحة الويب:

مــثـــال:

```
<!DOCTYPE html>
<html>
<body>

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
```

ملاحظة: تنتهي جمل PHP بفاصلة منقوطة.

الحساسية لحالة الأحرف

في PHP، لا توجد كلمات رئيسية حساسة لحالة الأحرف (مثل if وelse وwhile وecho، وغيرها)، والفئات والدوال والدوال المعرفة من قبل المستخدم.

في المثال التالي، جميع عبارات echo الثلاثة متساوية ومسموح بها:

مــثـــال:

```
<!DOCTYPE html>
<html>
<body>

<?php
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>

</body>
</html>
```

ملاحظة: جميع أسماء المتغيرات حساسة لحالة الأحرف!

انظر للمثال التالي؛ سيتم عرض قيمة الجملة الأولى `$color` فقط! هذا لأنه يتم التعامل مع `$color` و`$COLOR` و`$coLOR` كثلاثة متغيرات مختلفة:

مـثـــال:

```
<!DOCTYPE html>
<html>
<body>

<?php
$color = "red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>

</body>
</html>
```

تعليقات PHP

التعليق في أكواد PHP هو سطر لا يتم تنفيذه كجزء من البرنامج. الغرض الوحيد منه هو توضيح للشخص الذي ينظر إلى الكود.

يمكن استخدام التعليقات من أجل:

- دع الآخرين يفهمون كودك.
- ذكّر نفسك بما فعلته – عانى معظم المبرمجين من العودة إلى عملهم بعد عام أو عامين ولابد من إعادة فهم ما كتبوه. يمكن أن تذكرك التعليقات بما كنت تفكر فيه عندما كتبت الكود.

PHP تدعم عدة طرق للتعليق:

مــثـــال:

بنية التعليقات ذات السطر الواحد:

```
<?php
// This is a single-line comment
# This is also a single-line comment
?>
```

مـثـــال:

بنية التعليق متعدد الأسطر:

```
<?php
/*
This is a multiple-lines comment block
that spans over multiple
lines
*/
?>
```

مـثـــال:

استخدام التعليقات لإهمال جزء من الكود:

```
<?php
// You can also use comments to leave out
parts of a code line
$x = 5 /* + 15 */ + 5;
echo $x;
?>
```

المتغيرات

المتغيرات هي "حاويات" لتخزين المعلومات.

في PHP، يبدأ المتغير بعلامة $، متبوعًا باسم المتغير:

مــثـــال:

استخدام التعليقات لإهمال جزء من الكود:

```
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

بعد تنفيذ الجمل السابقة، سيحتفظ المتغير `$txt` بالقيمة `Hello world!` وسيحتفظ المتغير `$x` بالقيمة `5`، وسيحتفظ المتغير `$y` بالقيمة `10.5`.

ملاحظات:-

عندما تقوم بتعيين قيمة نصية لمتغير، ضع علامات اقتباس حول القيمة.

على عكس لغات البرمجة الأخرى، ليس لدى PHP أمر لإعلان المتغير. يتم إنشاؤه في اللحظة التي تحدد فيها القيمة لأول مرة.

يمكن أن يكون للمتغير اسم قصير (مثل x و y) أو اسم وصفي، مثل: (age، carname، total_volume).

قواعد متغيرات PHP:

- يبدأ المتغير بعلامة $، متبوعًا باسم المتغير.
- يجب أن يبدأ اسم المتغير بحرف أو شرطة سفلية (_).
- لا يمكن أن يبدأ اسم المتغير برقم.
- أسماء المتغيرات يمكن أن تحتوي فقط على أحرف أبجدية ورقمية وشرطات سفلية (A-z و 0-9 و _).
- أسماء المتغيرات حساسة لحالة الأحرف ($age و $AGE هما متغيرين مختلفين).

طباعة المتغيرات

غالبًا ما تستخدم جملة echo لإخراج البيانات على الشاشة.

يوضح المثال التالي كيفية إخراج النص والمتغير:

مـــثـــال:

```
<?php
$txt = "W3Schools.com";
echo "I love $txt!";
?>
```

المثال التالي سينتج نفس الناتج كما في المثال السابق:

مـــثـــال:

```
<?php
$txt = "W3Schools.com";
echo "I love " . $txt . "!";
?>
```

المثال التالي سيخرج مجموع المتغيرين:

مـــثـــال:

```
<?php
$x = 5;
$y = 4;
echo $x + $y;
?>
```

ملاحظة: سوف تتعلم المزيد عن جملة echo وكيفية إخراج البيانات على الشاشة في الفصل التالي.

في المثال السابق، لاحظ أننا لم نضطر إلى إخبار PHP بنوع المتغير.

يقوم PHP تلقائيًا بربط نوع البيانات بالمتغير، اعتمادًا على قيمته. نظرًا لعدم تعيين أنواع المتغيرات بالمعنى الدقيق للكلمة، يمكنك القيام بأشياء مثل إضافة سلسلة إلى عدد صحيح دون التسبب في خطأ.

في PHP 7، تمت إضافة تعريفات النوع. يوفر هذا خيارًا لتحديد نوع البيانات المتوقع عند الإعلان عن الدالة، وعن طريق تمكين طلب التدقيق (strict requirement)، سيؤدي ذلك إلى "خطأ فادح" في عدم تطابق النوع.

> **ملاحظة:** سوف تتعلم المزيد عن طلب strict وnon-strict أو التدقيق و عدم التدقيق، وتعريفات نوع البيانات في فصل دوال PHP.

نطاق متغيرات PHP

في PHP، يمكن الإعلان عن المتغيرات في أي مكان في البرنامج النصي.

نطاق المتغير: هو جزء من البرنامج النصي، ويحدد إين يمكن الرجوع إلى المتغير أو استخدامه.

يحتوي PHP على ثلاثة نطاقات مختلفة:

- **محلي**
- **عالمي**
- **ثابت**

النطاق العالمي والمحلي

المتغير المعرف خارج الدالة هو في النطاق العالمي، ولا يمكن الوصول إليه إلا خارج الدالة:

مـــثـــال:

متغير النطاق العالمي

```
<?php
$x = 5; // نطاق عالمي

function myTest() {
    // استخدام x داخل هذه الدالة سينتج خطأ //
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();

echo "<p>Variable x outside function is: $x</p>";
?>
```

المتغير الذي تم الإعلان عنه **داخل** الدالة هو في نطاق محلي ولا يمكن الوصول إليه إلا داخل هذه الدالة:

مــثــــال:

متغير النطاق المحلي

```
<?php
function myTest() {
    $x = 5; // local scope
    echo "<p>Variable x inside function is:
$x</p>";
}
myTest();

// استخدام x خارج هذه الدالة سينتج خطأ
echo "<p>Variable x outside function is:
$x</p>";
?>
```

يمكن أن يكون لديك متغيرات محلية بنفس الاسم في دوال مختلفة، لأن المتغيرات المحلية يتم التعرف عليها فقط داخل الدوال التي تم الإعلان فيها.

الكلمة الأساسية (العالمية "The global")

يتم استخدام الكلمة الأساسية `global` للوصول إلى متغير عالمي داخل دالة.

للقيام بذلك، استخدم الكلمة الأساسية `global` قبل المتغيرات (داخل الدالة):

مــثـــال:

```
<?php
$x = 5;
$y = 10;

function myTest() {
    global $x, $y;
    $y = $x + $y;
}

myTest();
echo $y; // outputs 15
?>
```

تقوم PHP أيضًا بتخزين جميع المتغيرات العالمية في مصفوفة تسمى `$GLOBALS[index]`. الـ *index* يحمل اسم المتغير. يمكن الوصول إلى هذه المجموعة أيضًا من داخل الدوال ويمكن استخدامها لتحديث المتغيرات العالمية مباشرةً.

يمكن إعادة كتابة المثال السابق كما يلي:

مـثـــال:

```
<?php
$x = 5;
$y = 10;

function myTest() {
    $GLOBALS['y'] = $GLOBALS['x']
+ $GLOBALS['y'];
}

myTest();
echo $y; // outputs 15
?>
```

الكلمة الأساسية (الثابت "The static")

عادة، عندما يتم الانتهاء من دالة أو تنفيذها، يتم حذف جميع المتغيرات. ومع ذلك، في بعض الأحيان نريد متغير محلي لا يحذف. نحن بحاجة إليه للحصول على دالة أخرى.

للقيام بذلك، نستخدم الكلمة الأساسية static عندما نقوم بتعريف المتغير لأول مرة:

مــثـــال:

```
<?php
function myTest() {
    static $x = 0;
    echo $x;
    $x++;
}

myTest();
myTest();
myTest();
?>
```

بعدها، في كل مرة يتم فيها استدعاء الدالة، سيظل هذا المتغير محتويا على المعلومات التي احتواها من آخر مرة تم فيها استدعاء الدالة.

ملاحظة: لا يزال المتغير محليًا في الدالة.

جملتي echo وprint

هناك طريقتان أساسيتان للإخراج: echo وprint.

في هذه الدروس التعليمية نستخدم echo أو print في كل مثال تقريبًا. لذلك، يحتوي هذا الفصل على مزيد من المعلومات حول جملتي الإخراج هاتين.

echo وprint هما نفس الشيء. كلاهما يستخدم لإخراج البيانات على الشاشة.

الاختلافات:

- echo لا ترجع قيمة، بينما print ترجع قيمة 1 بحيث يمكن استخدامها في التوضيحات.
- يمكن أن تحتوي echo على معلمات متعددة (على الرغم من أن هذا الاستخدام نادر الحدوث) بينما تأخذ print معلم واحد.
- echo أسرع من print.

جملة echo

يمكن استخدام جملة echo مع أو بدون أقواس: echo أو ()echo.

- **لعرض النص**

يوضح المثال التالي كيفية إخراج النص باستخدام أمر echo (لاحظ أن النص يمكن أن يحتوي على أوسمة HTML):

مـثــال:

```
<?php
echo "<h2>PHP is Fun!</h2>";
echo "Hello world!<br>";
echo "I'm about to learn PHP!<br>";
echo "هذا ", "النص ", "مكون ", "بأكثر من معلم.";
?>
```

- **لعرض قيم المتغيرات**

يوضح المثال التالي كيفية إخراج النص والمتغيرات باستخدام جملة echo:

مـثــــال:

```
<?php
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";
$x = 5;
$y = 4;
echo "<h2>" . $txt1 . "</h2>";
echo "Study PHP at " . $txt2 . "<br>";
echo $x + $y;
?>
```

جملة print

يمكن استخدام جملة print مع أو بدون أقواس: print أو ()print.

- **لعرض النص**

يوضح المثال التالي كيفية إخراج النص باستخدام أمر print (لاحظ أن النص يمكن أن يحتوي على أوسمة HTML):

مــثـــال:

```
<?php
print "<h2>PHP is Fun!</h2>";
print "Hello world!<br>";
print "I'm about to learn PHP!";
?>
```

- **لعرض المتغيرات**

يوضح المثال التالي كيفية إخراج النص والمتغيرات باستخدام جملة `print`:

مــثـــال:

```
<?php
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";
$x = 5;
$y = 4;

print "<h2>" . $txt1 . "</h2>";
print "Study PHP at " . $txt2 . "<br>";
print $x + $y;
?>
```

أنـواع بيـانـات PHP

يمكن للمتغيرات تخزين البيانات من أنواع مختلفة، وأنواع البيانات المختلفة يمكن أن تفعل أشياء مختلفة.

يدعم PHP أنواع البيانات التالية:

- سلسلة نصية "String"
- عدد صحيح "Integer"
- عدد كسري (ويسمى أيضًا مزدوج) "Float & double"
- منطقية "Boolean"
- مصفوفة "Array"
- كائن "Object"
- لا شيء "NULL"
- مورد "Resource"

السلسلة "String"

السلسلة عبارة عن سلسلة من الأحرف، مثل "Hello world!".

في السلسلة يمكنك وضع القيمة بين علامتي تنصيص مزدوجة أو فردية:

مـــثـــال:

```
<?php
$x = "Hello world!";
$y = 'Hello world!';

echo $x;
echo "<br>";
echo $y;
?>
```

الأعداد الصحيحة

نوع بيانات العدد الصحيح هو رقم غير عشري بين 2,147,483,648− و 2,147,483,647.

قواعد الأعداد الصحيحة:

- يجب أن يحتوي على رقم واحد على الأقل.
- يجب ألا يحتوي على علامة عشرية.
- يمكن أن يكون موجبا أو سالبا.
- يمكن تحديد الأعداد الصحيحة في: العشري (المستند على 10)، السادس عشري (المستند على 16) ، الثماني (المستند على 8)، أو الترميز الثنائي (المستند على 2).

في المثال التالي، $x هو عدد صحيح. ترجع الدالة `var_dump()` نوع البيانات والقيمة:

مـــثـــال:

```
<?php
$x = 5985;
var_dump($x); // int(5985)
?>
```

الأعداد الكسرية

الكسري: هو رقم ذو علامة عشرية أو رقم في الشكل الأسي.

في المثال التالي $x هو عدد كسري. ترجع الدالة var_dump() نوع البيانات والقيمة:

مــثــــال:

```
<?php
$x = 10.365;
var_dump($x);
?>
```

المنطقية

تمثل المنطقية حالتين محتملتين: TRUE أو FALSE.

```
$x = true;
$y = false;
```

غالبا ما تستخدم المنطقية في الاختبار الشرطي.

المصفوفات

تخزن المصفوفة قيم متعددة في متغير واحد.

في المثال التالي `$cars` عبارة عن مصفوفة. ترجع الدالة `var_dump()` نوع البيانات والقيمة:

مـــثـــال:

```
<?php
$cars = array("Volvo","BMW","Toyota");
var_dump($cars);
?>
```

الكائن

الكائن: هو نوع البيانات الذي يخزن البيانات والمعلومات حول كيفية معالجة تلك البيانات.

في PHP، يجب الإعلان عن الكائن.

أولا يجب أن نعلن صنف الكائن. لهذا، نستخدم الكلمة الأساسية الفئة "class".

الفئة (class): هي بنية يمكن أن تحتوي على خصائص وطرق:

مـــثـــال:

```
<?php
class Car {
    function Car() {
        $this->model = "VW";
    }
}

// create an object
$herbie = new Car();

// show object properties
echo $herbie->model;
?>
```

قيمة فارغة `null`

`Null`: هي نوع بيانات خاصة يمكن أن تحتوي على قيمة واحدة فقط: `NULL`.

متغير نوع البيانات `NULL` هو متغير ليس له قيمة مخصصة له.

نصيحة: إذا تم إنشاء متغير بدون قيمة، فإنه يتم تلقائيًا تعيينه كقيمة فارغة.

يمكن أيضًا إفراغ المتغيرات عن طريق تعيين القيمة إلى `NULL`:

مـــثـــال:

```
<?php
$x = "Hello world!";
$x = null;
var_dump($x);
?>
```

الموارد

نوع المورد خاص ليس نوع بيانات فعلي. إنه تخزين مرجع أو إشارة للدوال والموارد الخارجية لـ PHP.

مثال شائع لاستخدام نوع بيانات المورد، هو: استدعاء قاعدة بيانات.

لن نتحدث عن نوع المورد هنا، لأنه موضوع متقدم.

السـلا سل النصية لـPHP

وظائف السلسلة: في هذا الفصل، سنلقي نظرة على بعض الدوال شائعة الاستخدام لمعالجة السلاسل:

`strlen()` - إرجاع طول السلسلة

مــثـــال:

ارجاع طول السلسلة "Hello world!" :

```
<?php
echo strlen("Hello world!"); // outputs 12
?>
```

`str_word_count()` - حساب عدد الكلمات في السلسلة.

مــثـــال:

حساب عدد الكلمات في السلسلة "Hello world!":

```
<?php
echo str_word_count("Hello world!"); // outputs 2
?>
```

عكس السلسلة - strrev()

مــثـــال:

عكس سلسلة "Hello world!":

```
<?php
echo strrev("Hello world!"); // outputs !dlrow olleH
?>
```

البحث عن نص داخل سلسلة - strpos()

تبحث دالة strpos() عن نص معين داخل سلسلة. إذا تم العثور على مطابق، تقوم الدالة بإرجاع موضع أول حرف من المطابقة. إذا لم يتم العثور على تطابق، فسيتم إرجاع FALSE.

مــثـــال:

عكس السلسلة "Hello world!":

```
<?php
echo strpos("Hello world!", "world"); // outputs 6
?>
```

ملاحظة: الحرف الأول في السلسلة هو 0 (وليس 1).

str_replace() - استبدال نص داخل سلسلة

تقوم دالة `str_replace()` باستبدال بعض الأحرف ببعض الأحرف الأخرى في السلسلة.

مـــثـــال:

استبدال كلمة "world" بـ "Dolly":

```
<?php
echo str_replace("world", "Dolly", "Hello
world!"); // outputs Hello Dolly!
?>
```

الأرقـام

في هذا الفصل، سننظر بعمق في الأعداد الصحيحة، والكسرية، وسلاسل الأرقام.

أرقام PHP

PHP تحول نوع البيانات تلقائيا.

لذلك، إذا قمت بتعيين قيمة عدد صحيح لمتغير، فسيكون نوع هذا المتغير تلقائيًا عددًا صحيحًا. بعد ذلك، إذا قمت بتعيين سلسلة لنفس المتغير، فإن النوع سوف يتغير إلى سلسلة.

هذا التحويل التلقائي يمكنه في بعض الأحيان كسر الكود الخاص بك.

الأعداد الصحيحة Intergers

العدد الصحيح: هو رقم بدون أي جزء عشري.

2, 256, -256, 10358, -179567 كلها أعداد صحيحة، بينما 7.56, 10.0, 150.67 كسرية أو عشرية.

لذلك، نوع البيانات عدد–صحيح هو رقم غير عشري بين 2147483648– و 2147483647.
سيتم تخزين قيمة أكبر (أو أقل) من هذا، كما هو الكسري، لأنه يتجاوز الحد الأقصى للعدد الصحيح.

شيء مهم آخر يجب معرفته هو أنه حتى إذا كان ناتج عملية `2.5 * 4` هو `10`، فإن النتيجة يتم تخزينها بشكل كسري، لأن أحد المعاملات كان كسري (`2.5`).

فيما يلي بعض القواعد للأعداد الصحيحة:

- يجب أن يحتوي على رقم واحد على الأقل.
- يجب ألا يحتوي على علامة عشرية.
- يمكن أن يكون إيجابيا أو سلبيا.
- يمكن تحديد الأعداد الصحيحة في ثلاثة تنسيقات: عشري (يستند إلى 10) أو ست عشري (يستند إلى 16 – مسبوق بـ`0x`) أو ثماني (يستند إلى 8 – مسبوق بـ 0)

PHP لديها الدوال التالية للتحقق مما إذا كان نوع المتغير هو عدد صحيح:

- `is_int()`
- `is_integer() - alias of is_int()`
- `is_long() - alias of is_int()`

مـثـــال:

تحقق مما إذا كان نوع المتغير عددًا صحيحًا:

```
<?php
$x = 5985;
var_dump(is_int($x));

$x = 59.85;
var_dump(is_int($x));
?>
```

الأرقام الكسرية

الكسري: هو رقم بالعلامة العشرية أو رقم في شكل أسي.

2.0, 256.4, 10.358, 7.64E+5, 5.56E-5 كلها كسرية.

يمكن لنوع البيانات الكسري عادةً تخزين قيمة تصل إلى 1.7976931348623E+308 (تعتمد على النظام الأساسي)، ويبلغ الحد الأقصى 14 رقم.

يحتوي PHP على الدوال التالية للتحقق مما إذا كان نوع المتغير كسري:

- ❖ `is_float()`
- ❖ `is_double() - alias of is_float()`

مـــثـــال:

تحقق مما إذا كان نوع المتغير كسري:

```
<?php
$x = 10.365;
var_dump(is_float($x));
?>
```

ما لا نهاية

تعتبر القيمة الرقمية أكبر من PHP_FLOAT_MAX غير محدودة.

تحتوي PHP على الدوال التالية للتحقق مما إذا كانت القيمة الرقمية محدودة أو غير محدودة:

- `is_finite()`
- `is_infinite()`

ومع ذلك، ترجع الدالة `var_dump()` نوع البيانات والقيمة:

مــثـــال:

تحقق مما إذا كانت القيمة الرقمية محدودة أم لا نهائية:

```
<?php
$x = 1.9e411;
```

```
var_dump($x);
?>
```

NaN

NaN هي اختصار لـ "ليس رقما" "Not a Number".

تستخدم للعمليات الرياضية المستحيلة.

الدالة التالية للتحقق مما إذا كانت القيمة ليست رقمًا:

- ❖ is_nan()

ومع ذلك، ترجع الدالة var_dump() نوع البيانات والقيمة:

مــثـــال:

العلمية الغير صحيحة تخرج قيمة NaN:

```
<?php
$x = acos(8);
var_dump($x);
?>
```

السلاسل العددية

يمكن استخدام دالة `is_numeric()` لمعرفة ما إذا كان المتغير رقما. تُرجع الدالة `true` إذا كان المتغير رقمًا أو سلسلة رقمية، و `false` لغير ذلك.

مـــثـــال:

تحقق مما إذا كان المتغير رقميا:

```
<?php
$x = 5985;
var_dump(is_numeric($x));

$x = "5985";
var_dump(is_numeric($x));

$x = "59.85" + 100;
var_dump(is_numeric($x));

$x = "Hello";
var_dump(is_numeric($x));
?>
```

ملاحظة: من PHP 7.0: تقوم دالة `is_numeric()` بإرجاع FALSE للسلاسل الرقمية بالشكل السادس عشري (مثل 0xf4c3b00c)، حيث لم تعد تعتبر سلاسل رقمية.

ادخال سلاسل نصية وكسور على الأعداد الصحيحة

في بعض الأحيان تحتاج إلى وضع قيمة عددية في نوع بيانات آخر.

غالبًا ما تستخدم الدالة (`int`) أو (`integer`) أو `intval()` لتحويل القيمة إلى عدد صحيح.

مـثـــال:

وضع كسور وسلاسل في العدد الصحيح:

```
<?php
// Cast float to int
$x = 23465.768;
$int_cast = (int)$x;
echo $int_cast;

echo "<br>";

// Cast string to int
```

```
$x = "23465.768";
$int_cast = (int)$x;
echo $int_cast;
?>
```

الثوابت

الثوابت تشبه المتغيرات، إلا أنه بمجرد تعريفها، لا يمكن تغييرها أو عدم تعريفها.

الثابت هو معرف (اسم) لقيمة بسيطة. لا يمكن تغيير القيمة داخل البرنامج النصي.

يبدأ اسم الثابت بحرف أو شرطة سفلية (بدون علامة $ قبل الاسم الثابت).

> **ملاحظة:** على عكس المتغيرات، تكون الثوابت عالمية بشكل عام عبر البرنامج النصي بأكمله.

إنشاء ثابت

لإنشاء ثابت، استخدم الدالة `define()`.

بناء الجملة

```
define (الاسم, القيمة, غير حساسة لحالة الأحرف)
```

المعلمات:

- ❖ **الاسم:** اسم الثابت.
- ❖ **القيمة:** قيمة الثابت.
- ❖ **غير حساس لحالة الأحرف:** لتحديد ما إذا كان يجب أن يكون الاسم الثابت غير حساس لحالة الأحرف. الافتراضي `false`.

مــثـــال:

إنشاء ثابت باسم **حساس لحالة الأحرف:**

```
<?php
define("GREETING", "Welcome to W3Schools.com!");
echo GREETING;
?>
```

مــثـــال:

إنشاء ثابت باسم **غير حساس لحالة الأحرف:**

```
<?php
define("GREETING", "Welcome to W3Schools.com!", true);
echo greeting;
?>
```

مصفوفات ثابتة

في PHP7، يمكنك إنشاء مصفوفة ثابتة باستخدام دالة `define()`.

مـــثـــال:

إنشاء مصفوفة ثابتة:

```
<?php
define("cars", [
    "Alfa Romeo",
    "BMW",
    "Toyota"
]);
echo cars[0];
?>
```

ثوابت عالمية

الثوابت هي تلقائيا عالمية يمكن استخدامها عبر البرنامج النصي بأكمله.

مـــثـــال:

في هذا المثال ثابت داخل دالة، حتى إذا تم تعريفه خارج الدالة:

```
<?php
define("cars", [
```

```
    "Alfa Romeo",
    "BMW",
    "Toyota"
]);
echo cars[0];
?>
```

العمليات

يتم استخدام العمليات لإجراء عمليات على المتغيرات والقيم.

تقسم الـ PHP العمليات على مجموعات كالتالي:

- ❖ عمليات رياضية.
- ❖ عمليات التخصيص
- ❖ عمليات المقارنة.
- ❖ عمليات الزيادة والنقصان.
- ❖ عمليات منطقية.
- ❖ عمليات السلسلة.
- ❖ عمليات المصفوفة.
- ❖ علميات التخصيص الشرطي.

العمليات الرياضية

يتم استخدام العمليات الرياضية مع القيم الرقمية لأداء العمليات الحسابية المعروفة، مثل: الجمع والطرح والضرب إلخ.

العملية	الاسم	مثال
+	جمع	$x + $y
-	طرح	$x - $y
*	ضرب	$y * $x
/	قسمة	$y / $x
%	باقي القسمة	$y % $x
**	أس	$y ** $x

عمليات التخصيص

يتم استخدام عمليات التخصيص لوضع القيم الرقمية في المتغيرات.

عامل التخصيص الأساسي في PHP هو "=". وهذا يعني أن المعامل الأيسر يتم ضبطه على قيمة اليمين.

التخصيص	كمثل ..	وصف
x = y	x = y	سيتم وضع قيمة اليسار في قيمة اليمين
x += y	x = x + y	جمع
x -= y	x = x - y	طرح
x *= y	x = x * y	ضرب
x /= y	x = x / y	قسمة
x %= y	x = x % y	باقي القسمة

علميات المقارنة

تُستخدم عمليات المقارنة في PHP لمقارنة قيمتين (رقم أو سلسلة):

العملية	الاسم	مثال	الوصف
==	يساوي	`$x == $y`	سيرجع `true` إذا كان x يساوي y
===	يطابق	`$x === $y`	سيرجع `true` إذا كان x يساوي y، وكانوا من نفس النوع
!=	لا يساوي	`$x != $y`	سيرجع `true` إذا كان x لا يساوي y
<>	لا يساوي	`$x <> $y`	سيرجع `true` إذا كان x لا يساوي y
!==	لا يطابق	`$x !== $y`	سيرجع `true` إذا كان x لا يساوي y، أو لم يكونوا من نفس النوع
>	أكبر من	`$x > $y`	سيرجع `true` إذا كان x أكبر من y
<	أصغر من	`$x < $y`	سيرجع `true` إذا كان x أصغر منy
>=	أكبر من أو يساوي	`$x >= $y`	سيرجع `true` إذا كان x أكبر من أو يساوي y

`<=`	أصغر من أو يساوي	`$x <= $y`	سيرجع `true` إذا كان x أصغر من أو يساوي y
`<=>`	Spaceship	`$x <=> $y`	سيرجع رقم صحيح أقل من أو يساوي أو أكبر من الصفر، وهذا يتوقف على ما إذا كان `$x` أقل من أو يساوي أو أكبر من `$y`. قدم في PHP 7.

عمليات الزيادة والنقصان

يتم استخدام عمليات الزيادة لزيادة قيمة المتغير.

وتستخدم عمليات النقصان لتقليل قيمة المتغير.

العملية	الاسم	الوصف
`++$x`	الزيادة أولا	يزيد `$x` واحد ثم يرجع قيمة `$x`
`$x++`	الزيادة بعد	يرجع `$x` ثم يزيد واحد
`--$x`	النقص أولا	ينقص `$x` واحد، ثم يرجع قيمته
`$x--`	النقص بعد	يرجع قيمة `$x` ثم يزيدها واحد

العمليات المنطقية

يتم استخدام العمليات المنطقية لـ PHP لدمج الجمل الشرطية.

العملية	الاسم	مثال	الوصف
and	و	$x and $y	صحيح إذا كان $x و $y صحيحان
or	أو	$x or $y	صحيح إذا كان $x أو $y صحيح
xor		$x xor $y	صحيح إذا كان $x أو $y صحيحان، ولكن ليس كلاهما
&&	و	$x && $y	صحيح إذا كان $x و $y صحيحان
||	أو	$x || $y	صحيح إذا كان $x أو $y صحيح
!	ليس	!$x	صحيح إذا كان $x ليس صحيح

عمليات السلاسل النصية

تحتوي PHP على عاملين تم تصميمهما خصيصًا للسلاسل النصية.

العملية	الاسم	مثال	الوصف
.	تجميع	`$text1 . $text2`	جمع `$text1` و `$text2` (في الإخراج)
.=		`$text1 .= $txt2`	إضافة `$text2` لقيمة `$text1`

عمليات المصفوفة

يتم استخدام عمليات المصفوفة لمقارنة المصفوفات:

العملية	الاسم	مثال	الوصف
+	توحيد	`$x + $y`	توحيد `$x` و `$y`
==	مساواة	`$x == $y`	يرجع صح إذا كان `$x` و `$y` لهما نفس أزواج المفتاح/القيمة
===	مطابقة	`$x === $y`	يرجع صح إذا كان `$x` و `$y` لهما نفس أزواج المفاتيح/القيمة في نفس الترتيب ومن نفس النوع
!=	عدم المساواة	`$x != $y`	يرجع صح إذا كان `$x` لا يساوي `$y`
<>	عدم المساواة	`$x <> $y`	يرجع صح إذا كان `$x` لا يساوي `$y`
!==	عدم المطابقة	`$x !== $y`	يرجع صح إذا كان `$x` لا يطابق `$y`

عمليات التخصيص الشرطي

يتم استخدام عمليات التخصيص الشرطي لتخصيص قيمة وفقًا لشروط:

العملية	الاسم	مثال	الوصف
?:	ثلاثي "Ternary"	`$x = expr1 ? expr2 : expr3`	يرجع قيمة `$x`. قيمة `$x` هي expr2 إذا كانت expr1=TRUE. قيمة `$x` هي expr3 إذا كان expr1 = FALSE
??	Null coalescing	`$x = expr1 ?? expr2`	يرجع قيمة `$x`. قيمة `$x` هي expr1 في حالة وجود expr1، وهي ليست فارغة. إذا كان expr1 غير موجود أو كان NULL، فإن قيمة `$x` هي expr2. قدم في PHP 7

جمل if ... else ... elesif

تستخدم الجمل الشرطية لتنفيذ إجراءات مختلفة بناءً على شروط معينة.

في كثير من الأحيان عند كتابة التعليمات البرمجية، تريد تنفيذ إجراءات مختلفة لظروف معينة. يمكنك استخدام الجمل الشرطية في التعليمات البرمجية الخاصة بك للقيام بذلك.

في PHP لدينا العبارات الشرطية التالية:

- ❖ جملة `if` – ينفذ بعض الكود إذا كان هناك شرط واحد صحيح.
- ❖ جملة `if ... else` – ينفذ بعض الكود إذا كان الشرط صحيحا وكود آخر إذا كان هذا الشرط خاطئًا.
- ❖ جملة `if...elseif...else` – ينفذ الأكواد المختلفة لأكثر من شرطين.
- ❖ جملة `switch` – تختار واحدة من كتل الكود العديدة التي سيتم تنفيذها.

جملة if

جملة `if` تنفذ بعض التعليمات البرمجية إذا كان هناك شرط واحد صحيح.

بنية الجملة

```
if (الشرط ) {
    الكود المراد تشغيله إذا كان الشرط صحيح;
}
```

مـثــال:

إخراج "Have a good day!" إذا كان الوقت الحالي (الساعة) أقل من ٢٠:

```
<?php
$t = date("H");

if ($t < "20") {
```

```
    echo "Have a good day!";
}
?>
```

جملة الـ if ... else

الجملة if ... else تنفذ بعض الأكواد إذا كان الشرط صحيحًا وكود آخر إذا كان هذا الشرط خاطئًا.

بنية الجملة

```
if (الشرط ) {
    الكود المراد تشغيله إذا كان الشرط صحيحا;
} else {
    الكود المراد تشغيله إذا كان الشرط خاطئا;
}
```

مـــثـــال:

إخراج "Have a good day!" إذا كان الوقت الحالي أقل من 20، و "Have a good night!" لغير ذلك:

```
<?php
$t = date("H");
```

```
if ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
```

جملة if ... elseif ... else

بنية الجملة:

```
if (الشرط ) {
    الكود المراد تشغيله إذا كان الشرط صحيحا;
} elseif (الشرط ) {
    الكود المراد تشغيله إذا كان الشرط الأول خطأ وهذا الشرط صحيح;
} else {
    الكود المراد تشغيله إذا كانت الشروط كلها خطأ;
}
```

مــثـــال:

إخراج "Have a good day!" إذا كان الوقت الحالي أقل من 10، و "Have a good day!" إذا كان الوقت الحالي أقل من 20. غير ذلك سيخرج "Have a good night!":

```
<?php
$t = date("H");
if ($t < "10") {
```

```
    echo "Have a good morning!";
} elseif ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
```

جملة التبديل switch

يتم استخدام جملة switch لأداء إجراءات مختلفة بناءً على ظروف مختلفة.

استخدم جملة switch لاختيار أحد كتل التعليمات البرمجية العديدة المراد تنفيذها.

بناء الجملة

```
switch (n) {
    case label1:
        code to be executed if n=label1;
        break;
    case label2:
        code to be executed if n=label2;
        break;
    case label3:
        code to be executed if n=label3;
        break;
    ...
    default:
        code to be executed if n is different from
```

```
all labels;
}
```

هذه هي الطريقة التي تعمل بها: أولاً، لدينا تعبير واحد *n* (غالبًا يكون متغير)، يتم تقييمه مرة واحدة. ثم تتم مقارنة قيمة التعبير مع القيم لكل حالة في البنية. إذا كان هناك تطابق، يتم تنفيذ كتلة التعليمات البرمجية المرتبطة بهذه الحالة. استخدم `break` لمنع تشغيل الكود في الحالة التالية تلقائيًا. يتم استخدام الجملة `default` إذا لم يتم العثور على تطابق.

مـثـــال:

إخراج "Have a good day!" إذا كان الوقت الحالي أقل من 10، و "Have a good day!" إذا كان الوقت الحالي أقل من 20. غير ذلك سيخرج "Have a good night!":

```
<?php
$favcolor = "red";

switch ($favcolor) {
    case "red":
        echo "Your favorite color is red!";
        break;
    case "blue":
        echo "Your favorite color is blue!";
        break;
    case "green":
        echo "Your favorite color is
```

```
green!";
        break;
    default:
        echo "Your favorite color is neither
red, blue, nor green!";
}
?>
```

الحلقات

في الفصول التالية سوف تتعلم كيفية تكرار التعليمات البرمجية باستخدام حلقات PHP.

حلقات PHP

في كثير من الأحيان عند كتابة التعليمات البرمجية، تريد نفس كتلة التعليمات البرمجية مرارا وتكرارا لعدد معين من المرات. لذلك، بدلاً من إضافة العديد من أسطر التعليمات البرمجية متساوية تقريبًا في البرنامج النصي، يمكننا استخدام الحلقات.

يتم استخدام الحلقات لتنفيذ نفس الكود من التعليمات البرمجية مرارًا وتكرارًا، طالما كان هناك شرط معين صحيح.

في PHP، لدينا أنواع الحلقة التالية:

- **while**
- **do ... while**

- ❖ **for**
- ❖ **foreach** – تكرار كتلة أكواد لكل عنصر في المصفوفة.

> في الفصول التالية سوف نشرح ونعطي أمثلة على كل نوع.

الحلقة while – تكرار كتلة أكواد طالما أن الشرط المحدد صحيح.

بنية الجملة

```
while (condition is true) {
    code to be executed;
}
```

يعرض المثال التالي الأرقام من 1 إلى 5:

مـــثـــال:

```
<?php
$x = 1;

while($x <= 5) {
    echo "The number is: $x <br>";
    $x++;
```

```
}
?>
```

شرح المثال:

- ❖ `$x = 1;` – ضبط عداد الحلقة، اضبط قيمة البداية على 1.
- ❖ `$x <= 5` – استمر في الحلقة طالما أن `$x` أقل من أو تساوي 5.
- ❖ `$x++;` – قم بزيادة قيمة عداد الحلقة بمقدار ١ لكل تكرار.

يصل هذا المثال إلى 100 بواسطة عشرات:

مـــثـــال:

```
<?php
$x = 0;

while($x <= 100) {
    echo "The number is: $x <br>";
    $x+=10;
}
?>
```

شرح المثال:

- `$x = 0;` - ضبط عداد الحلقة (`$x`)، وتعيين قيمة البداية إلى 0.
- `$x <= 100` - استمر في الحلقة طالما `$x` أقل من أو يساوي 100.
- `$x+=10;` - زيادة عداد الحلقة بقيمة 10 لكل تكرار.

حلقة do while loop

حلقة `do … while` - يتم تنفيذ حلقة من التعليمات البرمجية مرة واحدة، ثم يكرر الحلقة طالما كان الشرط المحدد صحيحًا.

بنية الجملة

```
do {
    الكود الذي سينفذ;
} while (الشرط صحيح);
```

أمثلة:

المثال أدناه يعين أولاً متغير `$x` إلى 1 (`$x=1`). بعد ذلك، ستكتب حلقة التكرار بعض المخرجات، ثم تزيد المتغير `$x` بـ 1. ثم يتم التحقق من الشرط (يكون `$x` أقل من، أو يساوي 5؟)، وستستمر الحلقة في التشغيل طالما `$x` أقل من أو يساوي 5:

مـــثـــال:

```
<?php
$x = 1;
```

```
do {
    echo "The number is: $x <br>";
    $x++;
} while ($x <= 5);
?>
```

ملاحظة: في حلقة do...while، يتم اختبار الشرط بعد تنفيذ الجمل داخل الحلقة. هذا يعني أن حلقة do...while ستنفذ الجملة مرة واحدة على الأقل، حتى إذا كان الشرط خاطئ. انظر المثال التالي.

يعين هذا المثال المتغير $x على 6، ثم يقوم بتشغيل الحلقة، ثم يتم التحقق من الشرط:

مــثـــال:

```
<?php
$x = 6;

do {
    echo "The number is: $x <br>";
    $x++;
} while ($x <= 5);
?>
```

حلقة for

بناء الجملة

```
for (القيمة الإبتدائية ;اختبار القيمة ;الزيادة أو النقصان) {
    الكود المنفذ لكل دورة;
}
```

المعلمات:

- **القيمة الإبتدائية**
- **اختبار القيمة:** تقييم لكل تكرار. إذا تم تقييمه إلى TRUE، تستمر الحلقة. إذا تم تقييمه إلى FALSE، تنتهي الحلقة.
- **الزيادة أو النقصان:** يزيد من قيمة عداد الحلقة أو ينقص.

أمثلة:

يعرض المثال التالي الأرقام من 0 إلى 10:

مــثـــال:

```
<?php
for ($x = 0; $x <= 10; $x++) {
    echo "The number is: $x <br>";
}
?>
```

شرح المثال:

- ❖ `$x = 0;` – ضبط عداد الحلقة (`$x`)، وتعيين قيمة البداية إلى 0.
- ❖ `$x <= 10;` – استمرار الحلقة طالما `$x` أقل من أو يساوي 10.
- ❖ `$x++` – زيادة عداد الحلقة بعد كل تكرار.

مــثـــال:

```
<?php
for ($x = 0; $x <= 10; $x++) {
    echo "The number is: $x <br>";
}
?>
```

حلقة لكل foreach

حلقة foreach تعمل فقط على المصفوفات، ويتم استخدامها لإجراء حلقة داخل كل زوج مفتاح/قيمة في مصفوفة.

بناء الجملة

```
foreach ($مصفوفة as $قيمة ) {
  الكود الذي سينفذ;
}
```

لكل تكرار حلقة، يتم تعيين قيمة عنصر المصفوفة الحالية إلى $value ويتحرك مؤشر المصفوفة بواحد، حتى يصل إلى عنصر المصفوفة الأخير.

أمثلة

سيقوم المثال التالي بإخراج قيم المصفوفة المحددة (`$colors`):

مــثـــال:

```
<?php
$colors
= array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
  echo "$value <br>";
}
?>
```

المثال التالي سينتج المفاتيح والقيم الخاصة بالمصفوفة المحددة (`$age`):

مــثـــال:

```
<?php
$age
= array("Peter"=>"35", "Ben"=>"37", "Joe"=>"4
3");
foreach($age as $x => $val) {
  echo "$x = $val<br>";
}
?>
```

دوال PHP

القوة الحقيقية لـ PHP تكمن في دوالها.

تحتوي PHP على أكثر من 1000 دالة مدمجة، بالإضافة إلى ذلك، يمكنك إنشاء دوال مخصصة خاصة بك.

دوال PHP المعرفة من قبل المستخدم

إلى جانب دوال PHP المضمنة، من الممكن إنشاء دوالك الخاصة.

- ❖ الدالة: عبارة عن مجموعة من الجمل التي يمكن استخدامها بشكل متكرر في البرنامج.
- ❖ لن يتم تنفيذ الدالة تلقائيًا عند تحميل الصفحة.
- ❖ سيتم تنفيذ الدالة من خلال استدعائها.

إنشاء دالة معرفة من قبل المستخدم في PHP

يبدأ إعلان الدالة المعرفة من قبل المستخدم بكلمة `function`:

بناء الجملة

```
function اسم الدالة () {
    الكود المراد تنفيذه;
}
```

ملاحظة: يجب أن يبدأ اسم الدالة بحرف أو شرطة سفلية. أسماء الدوال ليست حساسة لحالة الأحرف.

نصيحة: أعطِ الدالة اسمًا يعكس وظيفتها!

في المثال السابق، نقوم بإنشاء دالة باسم "writeMsg()". يشير القوس المجعد المفتوح ({) إلى بداية رمز الدالة، ويشير القوس المجعد المغلق (}) إلى نهاية الدالة. مخرجات الدالة "Hello world!". لاستدعاء الدالة، فقط اكتب اسمها متبوعًا بالأقواس ():

مــثـــال:

```
<?php
function writeMsg() {
    echo "Hello world!";
}

writeMsg(); // call the function
?>
```

مدخلات الدوال

يمكن تمرير المعلومات إلى الدوال من خلال المدخلات. **المدخل** هو: تماما مثل المتغير.

يتم تعريف المدخلات بعد اسم الدالة، داخل الأقواس. يمكنك إضافة العديد من المدخلات كما تريد، والفصل بينها بفاصلة.

المثال التالي له دالة ذات مدخل واحد ($fname). عندما يتم استدعاء دالة familyName()، فإننا نمرر أيضًا اسمًا (مثل Jani)، ويتم استخدام الاسم داخل الدالة، مما يؤدي إلى إخراج عدة أسماء أولى مختلفة، ولكن اسماء أخيرة متساوية:

مـــثــــال:

```
<?php
function familyName($fname) {
    echo "$fname Refsnes.<br>";
}

familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
```

```
familyName("Borge");
?>
```

المثال التالي له دالة مع مدخلين (`$fname` و`$year`):

مــثـــال:

```
<?php
function familyName($fname, $year) {
    echo "$fname Refsnes. Born in $year
<br>";
}

familyName("Hege", "1975");
familyName("Stale", "1978");
familyName("Kai Jim", "1983");
?>
```

في المثال السابق، لاحظ أننا لم نضطر إلى إخبار PHP بنوع البيانات.

يقوم PHP تلقائيًا بربط نوع البيانات بالمتغير، اعتمادًا على قيمته. نظرًا لعدم تعيين أنواع البيانات بالمعنى الدقيق للكلمة، يمكنك القيام بأشياء مثل إضافة سلسلة إلى عدد صحيح دون التسبب في خطأ.

في PHP 7، تمت إضافة تعريفات النوع. هذا يعطينا خيارًا لتحديد نوع البيانات المتوقع عند الإعلان عن دالة، ومن خلال إضافة الإعلان (الصارم "strict")، سيؤدي ذلك إلى "خطأ فادح" في حالة عدم تطابق نوع البيانات.

في المثال التالي، نحاول إرسال رقم وسلسلة إلى الدالة دون استخدام التدقيق "strict":

مـــثـــال:

```
<?php
function addNumbers(int $a, int $b) {
    return $a + $b;
}
echo addNumbers(5, "5 days");
// since strict is NOT enabled "5 days" is
```

```
changed to int(5), and it will return 10
?>
```

لتمكين التدقيق strict، نحتاج إلى كتابة ; declare(strict_types=1). يجب أن يكون هذا في السطر الأول لملف الـ PHP.

في المثال التالي، نحاول إرسال رقم وسلسلة إلى الدالة، ولكننا قمنا بإضافة الإعلان strict:

مـــثـــال:

```
<?php declare(strict_types=1); // strict
requirement

function addNumbers(int $a, int $b) {
    return $a + $b;
}
echo addNumbers(5, "5 days");
// since strict is enabled and "5 days" is
not an integer, an error will be thrown
?>
```

الإعلان `strict` يفرض استخدام الأشياء بالطريقة المقصودة.

مدخلات قيم عشوائية

يوضح المثال التالي كيفية استخدام المدخل العشوائي. إذا قمنا باستدعاء الدالة `setHeight()` بدون مدخلات، فسوف تأخذ القيمة الافتراضية كمدخل:

مــثـــال:

```
<?php declare(strict_types=1); //طلب الـتـدقـيـق
function setHeight(int $minheight = 50) {
    echo "The height is : $minheight <br>";
}

setHeight(350);
setHeight(); // will use the default value of 50
setHeight(135);
setHeight(80);
?>
```

دوال – إرجاع القيم

للسماح للدالة بإرجاع القيم، استخدم جملة return:

مــثـــال:

```
<?php declare(strict_types=1); //طلب التدقيق
function sum(int $x, int $y) {
    $z = $x + $y;
    return $z;
}

echo "5 + 10 = " . sum(5, 10) . "<br>";
echo "7 + 13 = " . sum(7, 13) . "<br>";
```

```
echo "2 + 4 = " . sum(2, 4);
?>
```

دوال – إرجاع نوع المعرفات

تدعم PHP 7 أيضًا تعريفات النوع لجملة `return`. كما هو الحال مع إعلان النوع لمدخلات الدالة، من خلال تمكين التدقيق، سوف يظهر "خطأ فادح" عند عدم تطابق النوع.

لتحديد نوع لإرجاع الدالة، أضف نقطتين (:) والنوع مباشرة قبل قوس الفتح المجعد ({) عند تعريف الدالة.

في المثال التالي، نحدد نوع الإرجاع للدالة:

مــثـــال:

```
<?php declare(strict_types=1); //تمكين التدقيق
function addNumbers(float $a, float $b) :
float {
    return $a + $b;
}
echo addNumbers(1.2, 5.2);
?>
```

يمكنك تحديد أنواع إرجاع مختلفة عن أنواع المدخلات، ولكن تأكد من أن الإرجاع هو النوع الصحيح:

مـثـــال:

```
<?php declare(strict_types=1); // تمكين التدقيق
function addNumbers(float $a, float $b) :
int {
    return (int)($a + $b);
}
echo addNumbers(1.2, 5.2);
?>
```

المصفوفات

تخزن المصفوفة قيم متعددة في متغير واحد:

مــثـــال:

```
<?php
$cars      = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

ماهي المصفوفة؟

المصفوفة هي متغير خاص، يمكنها أن تحمل أكثر من قيمة في وقت واحد.

إذا كان لديك قائمة بالعناصر (قائمة بأسماء السيارات، على سبيل المثال)، فقد يبدو تخزين السيارات في متغيرات مفردة كما يلي:

```
$cars1 = "Volvo";
$cars2 = "BMW";
$cars3 = "Toyota";
```

ولكن؛ ماذا لو كنت ترغب في إنشاء حلقة للسيارات والعثور على واحدة محددة؟ وماذا لو لم يكن لديك 3 سيارات فقط، بل 300؟

الحل هو إنشاء مصفوفة!

يمكن أن تحتوي المصفوفة على العديد من القيم تحت اسم واحد، ويمكنك الوصول إلى القيم من خلال الرجوع إلى رقم الفهرس.

إنشاء مصفوفة

في PHP، يتم استخدام الدالة `array()` لإنشاء مصفوفة:

```
array();
```

في PHP، هناك ثلاثة أنواع من المصفوفات:

- **المصفوفات المفهرسة** – المصفوفات ذات فهرس رقمي.
- **المصفوفات الترابطية** – المصفوفات ذات المفاتيح المسماة.

❖ **مصفوفات متعددة الأبعاد** – مصفوفات تحتوي على مصفوفة واحدة أو أكثر.

للحصول على طول المصفوفة – نستخدم دالة الـ count()

يتم استخدام الدالة count() لإرجاع الطول (عدد العناصر) للمصفوفة:

مــثـــال:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo count($cars);
?>
```

المصفوفات المفهرسة

هناك طريقتان لإنشاء مصفوفات مفهرسة:

يمكن تعيين الفهرس تلقائيًا (يبدأ الفهرس دائمًا من 0)، مثل هذا:

```
$cars = array("Volvo", "BMW", "Toyota");
```

أو يمكن تعيين الفهرس يدويًا:

```
$cars[0] = "Volvo";
$cars[1] = "BMW";
$cars[2] = "Toyota";
```

يقوم المثال التالي بإنشاء مصفوفة مفهرسة باسم $cars، ويقوم بتعيين ثلاثة عناصر إليه، ثم يقوم بطباعة نص يحتوي على قيم المصفوفة:

مــثـــال:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1]
. " and " . $cars[2] . ".";
?>
```

حلقة بمصفوفة مفهرسة

للتنقل بين جميع قيم المصفوفة المفهرسة وطباعتها، يمكنك استخدام حلقة for، مثل هذه:

مــثـــال:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);

for($x = 0; $x < $arrlength; $x++) {
    echo $cars[$x];
    echo "<br>";
```

```
}
?>
```

المصفوفات الترابطية

المصفوفات المرتبطة: هي مصفوفات تستخدم مفاتيح معرفة تقوم بتعيينها لها.

هناك طريقتان لإنشاء مصفوفة مرتبطة:

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

أو:

```
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
```

يمكن بعد ذلك استخدام المفاتيح المعرفة في البرامج النصية:

مــثـــال:

```
<?php
$age
= array("Peter"=>"35", "Ben"=>"37", "Joe"=>"4
3");
echo "Peter is " . $age['Peter'] . " years
old.";
?>
```

حلقة داخل مصفوفة ترابطية لدخول وطباعة جميع قيم المصفوفة المترابطة، يمكنك استخدام حلقة foreach، مثل هذه:

مــثـــال:

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>
```

مصفوفات متعددة الأبعاد

في الصفحات السابقة، قمنا بوصف مصفوفات تمثل قائمة واحدة من أزواج مفاتيح/قيم.

ولكن، في بعض الأحيان تريد تخزين القيم باستخدام أكثر من مفتاح. لهذا، لدينا مصفوفات متعددة الأبعاد.

مصفوفة متعدد الأبعاد: عبارة عن مصفوفة تحتوي على مصفوفة واحدة أو أكثر.

تدعم PHP مصفوفات متعددة الأبعاد ذات مستويين أو ثلاثة أو أربعة أو خمسة أو أكثر بعمق. ومع ذلك، يصعب إدارة المصفوفات التي يزيد عمقها عن ثلاثة مستويات لأغلب الأشخاص.

يشير البعد الخاص بالمصفوفة إلى عدد المؤشرات التي تحتاجها لتحديد عنصر.

- لمصفوفة ثنائية الأبعاد، تحتاج إلى مؤشرين لتحديد عنصر.
- لمصفوفة ثلاثية الأبعاد، تحتاج إلى ثلاثة مؤشرات لتحديد عنصر.

مصفوفات ثنائية الأبعاد

مصفوفة ثنائية الأبعاد: هي مجموعة من المصفوفات (المصفوفة ثلاثية الأبعاد هي مجموعة من المصفوفات).

أولاً، ألقِ نظرة على الجدول التالي:

الاسم	الكمية	مباعة
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

يمكننا تخزين البيانات من الجدول أعلاه في مصفوفة ثنائية الأبعاد، مثل هذه:

```
$cars = array
  (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
  );
```

تحتوي مصفوفة $cars ثنائية الأبعاد الآن على أربعة مصفوفات، ولها مؤشران: صف وعمود.

للوصول إلى عناصر مصفوفة `$cars`، يجب أن نشير إلى المؤشرين (الصف والعمود):

مـــثـــال:

```
<?php
echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
?>
```

يمكننا أيضًا وضع حلقة `for` داخل حلقة `for` أخرى للحصول على عناصر مصفوفة `$cars` (لا يزال يتعين علينا الإشارة إلى المؤشرين):

مـــثـــال:

```php
<?php
for ($row = 0; $row < 4; $row++) {
  echo "<p><b>Row number $row</b></p>";
  echo "<ul>";
  for ($col = 0; $col < 3; $col++) {
    echo "<li>".$cars[$row][$col]."</li>";
  }
  echo "</ul>";
}
?>
```

فرز المصفوفات

يمكنك فرز العناصر في مصفوفة بترتيب أبجدي أو رقمي، تنازليًا أو تصاعديًا.

دوال لفرز المصفوفات

في هذا الفصل، سنتناول دوال فرز المصفوفات التالية:

الدالة	الوظيفة
sort()	فرز المصفوفات بترتيب تصاعدي
rsort()	فرز المصفوفات بترتيب تنازلي
asort()	فرز المصفوفات الترابطية بترتيب تصاعدي، حسب القيمة
ksort()	فرز المصفوفات الترابطية بترتيب تصاعدي، حسب المفتاح

`arsort()`	فرز المصفوفات الترابطية بترتيب تنازلي، حسب القيمة
`krsort()`	فرز المصفوفات الترابطية بترتيب تنازلي، حسب المفتاح

فرز المصفوفة بترتيب تصاعدي – `sort()`

يقوم المثال التالي بفرز عناصر المصفوفة `$cars` بترتيب أبجدي تصاعديا:

مــثـــال:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
sort($cars);
?>
```

يقوم المثال التالي بفرز عناصر مصفوفة `$numbers` بترتيب تصاعدي رقمي:

مـــثـــال:

```
<?php
$numbers = array(4, 6, 2, 22, 11);
sort($numbers);
?>
```

فرز المصفوفات بترتيب تنازلي – `rsort()`

المثال التالي يفرز عناصر مصفوفة `$cars` بالترتيب الأبجدي تنازليا:

مـــثـــال:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
rsort($cars);
?>
```

يقوم المثال التالي بفرز عناصر مصفوفة `$numbers` بالترتيب الرقمي تنازليا:

مـــثـــال:

```
<?php
$numbers = array(4, 6, 2, 22, 11);
rsort($numbers);
?>
```

فرز المصفوفة (ترتيب تصاعدي)، وفقًا للقيمة – asort()

يقوم المثال التالي بفرز مصفوفة مترابطة بترتيب تصاعدي، وفقًا للقيمة:

مـــثـــال:

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
asort($age);
?>
```

فرز المصفوفة (ترتيب تصاعدي)، وفقًا للمفتاح – ksort()

المثال التالي يفرز مصفوفة مترابطة بترتيب تصاعدي، وفقًا للمفتاح:

مــثـــال:

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
ksort($age);
?>
```

فرز مصفوفة (ترتيب تنازلي)، وفقا للقيمة – arsort()

مــثـــال:

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
arsort($age);
?>
```

فرز المصفوفة (ترتيب تنازلي)، وفقا للمفتاح – krsort()

المثال التالي يفرز مصفوفة مترابطة بترتيب تنازلي، وفقًا للمفتاح:

مــثـــال:

```
<?php
$age
= array("Peter"=>"35", "Ben"=>"37", "Joe"=>"4
3");
krsort($age);
?>
```

المتغيرات العالمية – Superglobals

تم تقديم Superglobals في PHP 4.1.0، وهي متغيرات مضمنة متوفرة دائمًا في جميع النطاقات.

المتغيرات العالمية – Superglobals

بعض المتغيرات المحددة مسبقًا في PHP هي "superglobals"، مما يعني أنه يمكن الوصول إليها دائمًا، بغض النظر عن النطاق – ويمكنك الوصول إليها من أي دالة أو فئة أو ملف دون الحاجة إلى القيام بأي شيء خاص.

المتغيرات الـ superglobal هي:

- ❖ $GLOBALS
- ❖ $_SERVER
- ❖ $_REQUEST
- ❖ $_POST
- ❖ $_GET
- ❖ $_FILES
- ❖ $_ENV
- ❖ $_COOKIE
- ❖ $_SESSION

المتغيرات العالمية الفائقة هي متغيرات مضمنة متوفرة دائمًا في جميع النطاقات.

$GLOBALS هو متغير عالمي فائق، يستخدم للوصول إلى المتغيرات العالمية من أي مكان في البرنامج النصي لـ PHP (أيضًا من داخل الدوال أو الطرق).

يخزن PHP جميع المتغيرات العامة في مصفوفة تسمى [index]$GLOBALS.

index يحمل اسم المتغير.

يوضح المثال التالي كيفية استخدام المتغير العالمي $GLOBALS:

مــثـــال:

```
<?php
$x = 75;
$y = 25;

function addition() {
    $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}

addition();
echo $z;
?>
```

$_SERVER

$_SERVER عبارة عن متغير super global يحتوي على معلومات حول الرؤوس والمسارات ومواقع البرامج النصية.

يوضح المثال التالي كيفية استخدام بعض العناصر في $_SERVER:

مــثـــال:

```
<?php
echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
echo "<br>";
echo $_SERVER['HTTP_HOST'];
echo "<br>";
echo $_SERVER['HTTP_REFERER'];
echo "<br>";
echo $_SERVER['HTTP_USER_AGENT'];
echo "<br>";
echo $_SERVER['SCRIPT_NAME'];
?>
```

يسرد الجدول التالي أهم العناصر التي يمكن إدخالها داخل $_SERVER:

إرجاع اسم ملف البرنامج النصي الذي يتم تنفيذه حاليًا:

```
1.    $_SERVER['PHP_SELF']
```

إرجاع إصدار واجهة البوابة العامة (CGI) "Common Gateway Interface" التي يستخدمها الخادم:

```
2.    $_SERVER['GATEWAY_INTERFACE']
```

إرجاع عنوان IP الخاص بالخادم المضيف:

```
3.    $_SERVER['SERVER_ADDR']
```

إرجاع اسم الخادم المضيف (مثل www.w3schools.com):

```
4.    $_SERVER['SERVER_NAME']
```

إرجاع سلسلة تعريف الخادم (مثل Apache/2.2.24):

```
5.    $_SERVER['SERVER_SOFTWARE']
```

إرجاع اسم ومراجعة بروتوكول المعلومات (مثل HTTP / 1.1):

```
6.    $_SERVER['SERVER_PROTOCOL']
```

إرجاع طريقة الطلب المستخدمة للوصول إلى الصفحة (مثل POST):

```
7.    $_SERVER['REQUEST_METHOD']
```

إرجاع الطابع الزمني لبداية الطلب (مثل 1377687496):

```
8.    $_SERVER['REQUEST_TIME']
```

إرجاع سلسلة الاستعلام إذا تم الوصول إلى الصفحة عبر سلسلة استعلام:

```
9.    $_SERVER['QUERY_STRING']
```

إرجاع رأس قبول من الطلب الحالي:

```
10.   $_SERVER['HTTP_ACCEPT']
```

إرجاع رأس Accept_Charset من الطلب الحالي (مثل UTF-8 و ISO-8859-1):

```
11.   $_SERVER['HTTP_ACCEPT_CHARSET']
```

إرجاع رأس المضيف من الطلب الحالي:

```
12.   $_SERVER['HTTP_HOST']
```

إرجاع عنوان URL الكامل للصفحة الحالية (غير موثوق لأنه لا يدعمها جميع وكلاء المستخدم):

```
13.   $_SERVER['HTTP_REFERER']
```

يتم الاستعلام عن البرنامج النصي من خلال بروتوكول HTTP الآمن:

```
14.   $_SERVER['HTTPS']
```

إرجاع عنوان IP من حيث يعرض المستخدم الصفحة الحالية:

```
15.   $_SERVER['REMOTE_ADDR']
```

إرجاع اسم المضيف من حيث يعرض المستخدم الصفحة الحالية:

```
16.   $_SERVER['REMOTE_HOST']
```

إرجاع المنفذ المستخدم على جهاز المستخدم للتواصل مع خادم الويب:

```
17.   $_SERVER['REMOTE_PORT']
```

إرجاع اسم المسار المطلق للبرنامج النصي الحالي:

```
18.   $_SERVER['SCRIPT_FILENAME']
```

تُرجع القيمة المعطاة إلى توجيه SERVER_ADMIN في ملف تكوين خادم الويب (إذا كان البرنامج النصي يعمل على مضيف افتراضي، فستكون القيمة المحددة لهذا المضيف الافتراضي) (مثل someone@w3schools.com):

```
19.   $_SERVER['SERVER_ADMIN']
```

إرجاع المنفذ على جهاز الخادم الذي يستخدمه خادم الويب للاتصال (مثل 80):

```
20.   $_SERVER['SERVER_PORT']
```

إرجاع إصدار الخادم واسم المضيف الافتراضي الذي تمت إضافته إلى الصفحات التي أنشأها الخادم:

```
21.   $_SERVER['SERVER_SIGNATURE']
```

إرجاع المسار المستند لنظام الملفات إلى البرنامج النصي الحالي:

```
22.    $_SERVER['PATH_TRANSLATED']
```

إرجاع مسار البرنامج النصي الحالي:

```
23.    $_SERVER['SCRIPT_NAME']
```

إرجاع عنوان URI للصفحة الحالية:

```
24.    $_SERVER['SCRIPT_URI']
```

$_REQUEST

$ _REQUEST : يستخدم لجمع البيانات بعد تقديم نموذج HTML.

يوضح المثال التالي نموذجًا يحتوي على حقل إدخال وزر إرسال. عندما يرسل المستخدم البيانات بالنقر فوق "submit"، يتم إرسال بيانات النموذج إلى الملف المحدد في خاصية الإجراء "action" للوسم <form>. في هذا المثال، نشير إلى هذا الملف نفسه لمعالجة بيانات النموذج. إذا كنت ترغب في استخدام ملف PHP آخر لمعالجة بيانات النموذج، فاستبدل ذلك باسم الملف الذي تختاره. بعد ذلك، يمكننا استخدام $_REQUEST لجمع قيم حقل الإدخال:

مـثــال:

```
<html>
<body>
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
```

```
  Name: <input type="text" name="fname">
  <input type="submit">
</form>
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // collect value of input field
    $name = $_REQUEST['fname'];
    if (empty($name)) {
        echo "Name is empty";
    } else {
        echo $name;
    }
}
?>
</body>
</html>
```

$_POST

$_POST يُستخدم لجمع بيانات النموذج بعد إرسال نموذج HTML من خلال method = "post". يستخدم $_POST أيضًا على نطاق واسع لتمرير المتغيرات.

يوضح المثال التالي نموذجًا يحتوي على حقل إدخال وزر إرسال. عندما يرسل المستخدم البيانات بالنقر فوق "إرسال"، يتم إرسال بيانات النموذج إلى الملف المحدد في خاصية الإجراء للوسم <form>. في هذا المثال، نشير إلى الملف نفسه لمعالجة بيانات النموذج. إذا كنت ترغب في استخدام ملف PHP آخر لمعالجة بيانات النموذج، فاستبدل ذلك باسم الملف الذي تختاره. بعد ذلك، يمكننا استخدام $_POST لجمع قيمة حقل الإدخال:

مـثــال:

```
<html>
<body>
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
```

```
  Name: <input type="text" name="fname">
        <input type="submit">
</form>
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // collect value of input field
    $name = $_POST['fname'];
    if (empty($name)) {
        echo "Name is empty";
    } else {
        echo $name;
    }
}
?>
</body>
</html>
```

$_GET

$_GET: تستخدم لجمع بيانات النموذج بعد إرسال نموذج HTML بـ "method = "get.

يمكنه أيضًا جمع البيانات المرسلة في عنوان URL.

افترض أن لدينا صفحة HTML تحتوي على ارتباط تشعبي مع معلمات:

```
<html>
<body>

<a href="test_get.php?subject=PHP&web=W3schools.com">Test $GET</a>

</body>
</html>
```

عندما ينقر المستخدم على الرابط "Test GET"، يتم إرسال المعلمتين "topic" و "web" إلى "test_get.php"، ثم يمكنك الوصول إلى قيمهما في "test_get.php" بـ $_GET.

المثال التالي يوضح الكود الموجود في "test_get.php":

مـــثـــال:

```
<?php
echo "Study " . $_GET['subject'] . " at
" . $_GET['web'];
?>
```